LAB 5

1) create a namespace iti-devops

```
controlplane $ kubectl create namespace iti-devops
namespace/iti-devops created
controlplane $
```

2) create a service account iti-sa-devops under the same namespace

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: iti-sa-devops
  namespace: iti-devops
```

```
controlplane $ kubectl apply -f service.yaml
serviceaccount/iti-sa-devops created
```

3) create a clusteRole which should be named as cluster-role-devops to grant permissions “get”,”list”,”watch”,”create”,”patch”,”update” to “configMaps”,”secrets”,”endpoints”,”nodes”,”pods”,”services”,”namespaces”,”events”,”serviceAccounts”.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: cluster-role-devops
rules:
- apiGroups: [""]
  resources: ["services", "endpoints", "pods", "configMaps", "secrets", "nodes", "namespaces", "events", "serviceAccounts"]
  verbs: ["get", "list", "watch", "create", "patch", "update"]
```

```
clusterrole.rbac.authorization.k8s.io/cluster-role-devops created
controlplane $
```

4) create a ClusterRoleBinding which should be named as cluster-role-binding-devops under the same namespace. Define roleRef apiGroup should be rbac.authorization.k8s.io. Kind should be ClusterRole, name should be cluster-role-devops and subjects kind should be ServiceAccount: name should be iti-sa□devops and namespace should be iti-devops

```
controlplane $ kubectl apply -f clustrole-bind.yaml
clusterrolebinding.rbac.authorization.k8s.io/cluster-role-binding-devops created
controlplane $
```

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: cluster-role-binding-devops
subjects:
- kind: ServiceAccount
  name: iti-sa-devops
  namespace: iti-devops
roleRef:
  kind: ClusterRole
  name: cluster-role-devops
  apiGroup: rbac.authorization.k8s.io
```

5) What is the difference between statefulSets and deployments?

>> Statefulsets: A StatefulSet is a workload API object for managing stateful applications. Usually, Kubernetes users are not concerned with how pods are scheduled, although they do require pods to be deployed in order, to be attached to persistent storage volumes, and to have unique, persistent network IDs that are retained through rescheduling. StatefulSets can help achieve these objectives. Like Deployments, StatefulSets manage the pods based on the same container specifications. However, they differ from deployments in that they maintain sticky identities for each pod. Pods may be created from an identical spec, but they are not interchangeable and are thus assigned unique identifiers that persist through rescheduling.

>> Deployments: A Deployment is a Kubernetes resource object used for declarative application updates. Deployments allow you to define the lifecycle of applications, including the container images they use, the number of pods and the manner of updating them. Deployments are fully managed by the backend in Kubernetes, with the entire update process being server side, with no client involvement. They ensure that a specified number of pods are always running and available. The entire update process is recorded, with versioning to provide options for pausing, resuming or rolling back to previous versions.

6) Set up Ingress on Minikube with the NGINX Ingress Controllerplay around with paths , you can create more than 2 deployments if you like
https://kubernetes.io/docs/tasks/access-application-cluster/ingress-minikube/

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: example-ingress
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /$1
spec:
  rules:
    - host: hello-world.info
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: web2
                port:
                  number: 8080
          - path: /v2
            pathType: Prefix
            backend:
              service:
                name: web2
                port:
                  number: 8080
```